2013 Eighth Asia Joint Conference on Information Security

Formal Analysis of ISO/IEC 9798-2 Authentication Standard using AVISPA

Sheikh Ziauddin
Department of Computer Science
COMSATS Institute of Information Technology
Park Road, Islamabad, Pakistan
Email: sheikh.ziauddin@comsats.edu.pk

Bruno Martin
Université Nice Sophia Antipolis,
I3S-UMR 7271 CNRS, BP 121,
F-06903 Sophia Antipolis cedex, France
Email: Bruno.Martin@unice.fr

***Abstract*—Use of formal methods is considered as a useful and efficient technique for the validation of security properties of the protocols. In this paper, we analyze the protocols of ISO/IEC 9798-2 entity authentication standard using a state-of-the-art tool for automated analysis named AVISPA. Our analysis of the standard using AVISPA's OFMC and CL-AtSe back-ends shows that the two party protocols are secure against the specified security properties while the back-ends are able to find attacks against unilateral and mutual authentication protocols involving a trusted third party.**

I. Introduction

One of the main goals of cryptography is entity authentication. Entity authentication is the process of authenticating/verifying entity for what it claims to be. A lot of research has been done and many authentication protocols have been presented by many researchers. Lamport [1] presented one of the earliest password-based entity authentication schemes in 1981. In Lamport's scheme, password tables were stored on the server and a compromise of the server lead to severe security risks. Subsequently, many authentication schemes [2], [3], [4], [5], [6], [7], [8] have been proposed to achieve better security and efficiency.

Traditionally, a typical cycle has been followed for designing security protocols. This cycle includes, specifying the requirements of the protocol, designing the protocol according to the given specifications, manual analysis of the protocol to find weaknesses, improving or redesigning the protocols by removing the flaws, then again analyzing the new protocol to find flaws and so on. This approach has problems because protocols are analyzed manually and manual analysis is slow, tiresome and error prone. For example, Hwang et al. [9] presented a two factor mutual authentication protocol but Yoon et al. [10] analyzed their protocol and pointed that it provides only unilateral authentication and is also vulnerable to denial of service attack. Yoon et al. also came up with an improved redesigned protocol. Later on, Chun-Ta Li et al. [11] showed that improved scheme presented by Yoon et al. is still unable to provide guard against denial of service attack and it still provides only unilateral authentication. Later, Kalsoom et al. [12] found that Chun-Ta Li's improved protocol is susceptible to impersonation attack if smart card is lost. Therefore, the process of manually analyzing the protocols is not only time consuming but also costly once design errors are found in a protocol after it has been deployed in a practical setting.

Due to the above mentioned issues with manual design and analysis of security protocols, research community started working on formal analysis of security protocols. Two different approaches have been used for automated designing/analysis of security protocols:

- The first approach is due to Abadi and Needham [13]. The idea is to use a design methodology which will result in secure protocol design. They presented principles for better and secure design of security protocols. Though the presented principles are not necessary and sufficient conditions for protocol correctness, they provide an informal guideline to avoid many common design errors.
- The second approach is to use systematic means for analysis after the protocols are designed. The analysis contains either finding security flaws in design or proving correctness of certain security properties. Under this approach, the most well-known instance is due to Burrows et al. [14]. They presented logic to analyze the security properties of protocols. Using this strategy, they analyze protocols to answer questions about their correctness, output, assumptions and whether they perform any irrelevant operations. They define some logical postulates and rules of knowledge and belief to analyze protocol and show effectiveness of their logic by analyzing different protocols from the existing literature.

Most of the modern automated protocol analysis tools are based on model-checking techniques. The use of model-checkers for security protocols became popular when Lowe [15] used FDR model-checker to find a security flaw in Needham-Schroeder public key protocol. Mur-phi [16] is another model-checker designed for the automatic protocol analysis where user has to specify the protocol and desired security requirements using *Mur-phi* language. Then the protocol is searched against specified security requirements using either depth-first search or breadth-first search strategy. Later, Song et al. presented Athena [17] that was an extension of the Strand Space Model where logic is integrated to specify security properties and an automatic procedure is used for the evaluation of well formed formulae. After the evaluation

This work was supported by The French National Research Agency, project EMC (ANR-09-BLAN-0164).

978-0-7695-5075-6/13 $26.00 © 2013 IEEE
DOI 10.1109/ASIAJCIS.2013.25

of formula, it provides a proof if formula is evaluated as true and shows a counter example if formula is evaluated as false. More recently, Cremers [18] presented one of the state of the art freely available tools Scyther. It supports multi-protocol analysis and analyzes infinite number of sessions to verify the correctness of protocols. The protocol is specified in SPDL language. Later on, Basin and Cremers [19] performed a formal analysis of ISO/IEC 9798 standard using Scyther and found a number of weaknesses in some of the protocols in the family. AVISPA [20], [21] is another modern tool for automatic analysis of protocols which we have used to analyze ISO/IEC 9798-2 authentication standard in this paper. AVISPA is considered robust due to its modular approach and it is capable of analyzing large industrial scale real world protocols. The protocols are specified using HLPSL language.

The rest of this paper is organized as follows. Section II gives an overview of AVISPA tool. ISO/IEC 9798-2 authentication standard is described in Section III. Formal analysis of the standard using AVISPA is detailed in Section IV while Section V summarises our conclusions.

II. The AVISPA Tool

AVISPA (Automated Validation of Internet Security Protocols and Applications) is a modern tool used for automated *falsification* of security protocols. Protocol *falsification* and protocol *verification* differ in the fact that in falsification, the tool tries to detect attacks against the protocols being tested whereas in verification, the tool tries to prove the correctness of the protocols being tested. AVISPA uses a special modelling language named High Level Protocol Specification Language (HLPSL) to write specifications for security protocols. These high level specifications written in HLPSL are translated to low level Intermediate Format (IF) using an *hlpsl2if* translator. The low level intermediate format code is then executed by one of the four supported back-ends in order to find any potential security vulnerabilities in the protocols being analysed.

AVISPA is quite robust as it uses a modular approach where the back-end tools are independent of the specification language. The approach is quite similar to that of Java where the high level Java code is translated to low level byte-code which is platform independent and can be executed on any machine. Similarly, separation of back-end tools from the specification language gives the flexibility to AVISPA that third parties can develop their own back-ends as long as they follow the intermediate format.

There are a number of tools in the market which provide automated falsification of security protocols but unfortunately most of the tools work well only for small to medium sized protocols. AVISPA, on the other hand, is capable of analysing large industrial scale real world protocols. It includes a large library of security protocols' specifications collection containing many large protocols as well. The current library contains 33 protocols with 215 security properties (each security goal potentially met by a protocol is treated as a separate problem resulting in one protocol having multiple security problems).

The architecture of AVISPA tool is shown in Figure 1. Currently, AVISPA supports four back-ends: OFMC, CL-AtSe, SATMC and TA4SP. OFMC (On-the-Fly Model Checker) considers both typed and untyped protocol models and carries

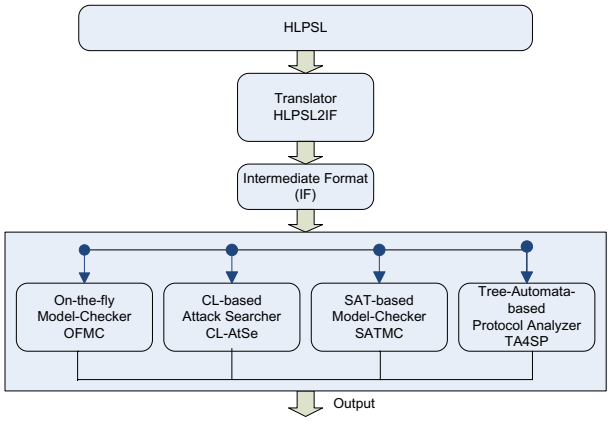

Fig. 1: The AVISPA Architecture.

out on-the-fly protocol falsification and bounded verification. It uses lazy intruder technique for the creation of efficient infinite state space models and incorporates symbolic techniques to model Dolev-Yao intruder. CL-AtSe (Constraint-Logic-based Attack Searcher) also works on typed and untyped protocol models and performs protocol falsification and bounded verification. It incorporates various optimization techniques to reduce redundancies and uselessness in the protocol. SATMC (SAT-based Model Checker) performs protocol falsification and bounded verification for only typed protocol model. SATMC works by reducing the security problem of a protocol to satisfiability of a propositional formula which is solved by state-of-the-art SAT solvers. Lastly, TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) is used for the unbounded verification of security properties of the protocol. It uses rewriting rules to represent protocol and intruders abilities and tree language is used to map the intruder's knowledge and instances of sessions. Then set of reachable terms is computed from the tree automaton language using rewriting rules.

A. High Level Protocol Specification Language

HLPSL is a high level language used to specify the protocols and their security properties in AVISPA. HLPSL is a role-based language where each participant is represented as a role. There are two types of roles; basic roles and composed roles. A basic role shows the actions of a single participant during the execution of a protocol or a sub-protocol while a composed role combines two or more basic roles to execute them together. The definition of a basic role represents the initial knowledge, initial state and the transitions of that role. The general structure to define a basic role is given below.

```
role name_of_the_role (arguments)
played_by role_identifier
def=
   local
   State: type,
   ..
   ..
init
   State:=initial_value
```

```
    ..
    ..
transition
    ..
    ..
end role
```

where *arguments* is the list of arguments showing the initial knowledge of the role, *local* is a keyword under which local variables of the role are defined, and under *init* section, the local variables are initialized. Under *transition* section, role transitions in terms of state changes are defined. The syntax of composed roles is quite similar to that of basic roles. The only difference is presence of *composition* section in composed roles instead of basic roles' *transition* section.

After roles definitions, another important part of HLPSL is declaration of goals. In *goal* section, security properties are specified against which the protocol is analyzed. If any of the goals is violated, the AVISPA back-ends report an attack against the protocol along with a complete trace of that attack. There are two types of security properties (goals) which can directly be specified using HLPSL. The confidentiality goal(s) are specified using the keyword *secrecy_of* while authentication goal(s) use *authentication_on* keyword. A sample goal section is given below.

```
goal
   secrecy_of k
   authentication_on alice_bob_na
end goal
```

After specifying the roles and declaring the goals, the last thing is to execute the code by calling the highest level composite role *environment*.

III. ISO/IEC 9798-2

ISO-IEC 9798 is a set of protocol families produced by joint technical committee of ISO and IEC for entity authentication. ISO-IEC 9798 consists of 6 parts. ISO-IEC 9798-1 presents an authentication model as well as general requirements and constraints for entity authentication. ISO-IEC 9798-2 to ISO-IEC 9798-5 provide specific mechanisms for entity authentication using symmetric encipherment, digital signatures, cryptographic check functions and zero knowledge techniques, respectively. The last part, ISO-IEC 9798-6, deals with manual techniques for data transfer between authenticating devices.

In this paper, we formally analyse part 2 of ISO-IEC 9798 standard [22] namely *mechanisms using symmetric encipherment algorithms*. AVISPA tool [20], [21] is used to analyse these protocols. ISO-IEC 9798-2 contains 6 different mechanisms for entity authentication. First two mechanisms provide unilateral authentication in a two-party setting, next two provide mutual authentication in a two-party setting while mechanisms five and six provide mutual authentication (or unilateral authentication by omitting a couple of steps) in a three-party setting involving an additional trusted third party.

Before describing these mechanisms, we want to make two remarks regarding our specifications and analysis of these mechanisms.

Notation	Meaning
K_{ab}	Symmetric key shared between Alice and Bob
K_{at}	Symmetric key shared between Alice and trusted third party
K_{bt}	Symmetric key shared between Bob and trusted third party
N_a	Sequence number or time stamp generated by Alice
N_b	Sequence number or time stamp generated by Bob
N_t	Sequence number or time stamp generated by trusted third party
A	Identification of Alice
B	Identification of Bob
$\mathsf{E}_{\mathsf{K_{ab}}}(X)$	Encipherment of X using the symmetric key K_{ab}
$X\|\|Y$	Concatenation of strings X and Y

TABLE I: Notations used in this paper to describe ISO/IEC 9798-2 standard

- Different mechanisms in 9798-2 use some optional text fields. We have ignored these fields and not used these in our analysis.
- As mentioned earlier, there are two mechanisms for unilateral authentication in two-party setting. The first mechanism is a one-pass authentication mechanism which uses sequence numbers or time stamps in order to provide security against replay attacks. To avoid the same attack, the second mechanism uses random numbers in a challenge-response style requiring two messages to be passed for unilateral authentication. Similarly, for mutual authentication (in two-party as well as three-party setting), one additional message is passed when using random numbers as compared to the use of sequence numbers or time stamps. In our analysis, we consider only those mechanisms that involve sequence number/time stamp to avoid replay attack and we model these sequence numbers/time stamps as *nonce* data types in our HLPSL specifications. Therefore, this paper models and analyses four protocols in ISO-IEC 9798-2.

In the following sections, we describe these mechanisms one by one. The notations we have used to denote different elements are shown in Table I.

A. Mechanisms Not Involving a Trusted Third Party

This scenario involves Alice and Bob who share a symmetric key in advance. In first mechanism, Alice is authenticated by Bob while in the second mechanism, both Alice and Bob authenticate each other.

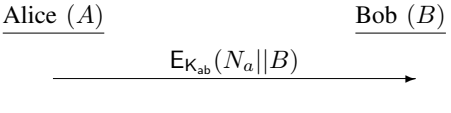

Fig. 2: Message passed between Alice and Bob for unilateral authentication

1) Unilateral Authentication: In unilateral authentication mechanism described in ISO-IEC 9798-2 standard, the claimant Alice initiates the communication sending a single message to the verifier Bob who authenticates Alice. The following steps are involved.

- Alice sends the value $\mathsf{E}_{\mathsf{K_{ab}}}(N_a||B)$ to Bob.
- Upon receiving the message, Bob decrypts $\mathsf{E}_{\mathsf{K_{ab}}}(N_a||B)$ and checks the correctness of identifier B and the nonce N_a.

Figure 2 gives a graphical representation of unilateral authentication mechanism.

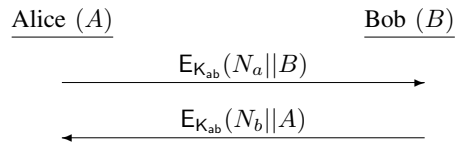

Fig. 3: Messages passed between Alice and Bob for mutual authentication

2) Mutual Authentication: In mutual authentication mechanism, Alice initiates the communication sending the first message to Bob who authenticates Alice. Next, Bob sends a message back to Alice who authenticates Bob. The following steps are involved.

- Alice sends the value $\mathsf{E}_{\mathsf{K_{ab}}}(N_a||B)$ to Bob.
- Upon receiving the message, Bob decrypts $\mathsf{E}_{\mathsf{K_{ab}}}(N_a||B)$ and checks the correctness of identifier B and the nonce N_a.
- Bob sends the value $\mathsf{E}_{\mathsf{K_{ab}}}(N_b||A)$ to Alice.
- Upon receiving the message, Alice decrypts $\mathsf{E}_{\mathsf{K_{ab}}}(N_b||A)$ and checks the correctness of identifier A and the nonce N_b.

Mutual authentication mechanism in ISO/IEC 9798-2 is graphically represented in Figure 3.

B. Mechanisms Involving a Trusted Third Party

In this scenario, Alice and Bob do not share any symmetric key among themselves in advance. Instead, both share symmetric keys with the trusted third party. The symmetric session key K_{ab} to be shared between Alice and Bob is generated by trusted third party (TTP) and is sent to Alice and Bob in a digital envelope.

1) Unilateral Authentication: In three-party unilateral authentication mechanism, the claimant Alice initiates the communication with trusted third party. Alice, in turn, receives an encrypted message from TTP, compiles a new message using the message received from TTP and send this new message to Bob. Bob authenticates Alice using the received message. The following steps are involved.

- Alice sends the value $N_{a1}||B$ to TTP where N_{a1} is a nonce generated by Alice.
- TTP sends the value $\mathsf{E}_{\mathsf{K_{at}}}(N_{a1}||K_{ab}||B)||$ $\mathsf{E}_{\mathsf{K_{bt}}}(N_t||K_{ab}||A)$ to Alice.
- Upon receiving the message, Alice decrypts the first part of the received message, i.e., $\mathsf{E}_{\mathsf{K_{at}}}(N_{a1}||K_{ab}||B)$, checks the correctness of identifier B, verifies that the nonce N_{a1} corresponds to the one sent to TTP in the first message and retrieves the key K_{ab}. She also extracts the second part of the received message containing the digital envelope.
- Alice sends the message $\mathsf{E}_{\mathsf{K_{bt}}}(N_t||K_{ab}||A)||$ $\mathsf{E}_{\mathsf{K_{ab}}}(N_{a2}||B)$ to Bob.
- Upon receiving the message $\mathsf{E}_{\mathsf{K_{bt}}}(N_t||K_{ab}||A)||$ $\mathsf{E}_{\mathsf{K_{ab}}}(N_{a2}||B)$, Bob decrypts the message, retrieves the secret key K_{ab} and verifies the correctness of identifiers A and B along with the nonces N_t and N_{a2}.

Figure 4 gives a graphical representation of unilateral authentication mechanism involving a trusted third party.

2) Mutual Authentication: In ISO/IEC 9798-2 standard, three-party mutual authentication mechanism is based on three-party unilateral authentication mechanism. In mutual authentication, the first three messages to be passed are identical to unilateral authentication version. There is one additional message sent by Bob to Alice as shown in Figure 5. The additional steps to the unilateral authentication mechanism are the following.

- Bob sends the value $\mathsf{E}_{\mathsf{K_{ab}}}(N_b||A)$ to Alice.
- Upon receiving the message, Alice decrypts it and checks the correctness of identifier A and the nonce N_b.

IV. Formal Analysis using AVISPA

As stated earlier, we have analysed four protocols of ISO-IEC 9798-2 standard. We used OFMC and CL-AtSe model analysers for analysis of each protocol. The first 2 protocols (i.e., the ones without TTP) were declared secure by both OFMC and CL-AtSe. On the other hand, both OFMC and CL-AtSe found attacks against the last two protocols (the ones with TTP). As detected attacks against both protocols are quite similar, here we provide details related to one of those protocols namely unilateral authentication protocol involving a trusted third party (Figure 4). Our specification for this protocol contains 3 roles namely *alice, bob* and *thirdparty*. An excerpt from our HLPSL code containing the role *alice* is shown below:

```
role alice(
   A,B,TTP : agent,
   Kat : symmetric_key,
   ST,RT,SB,RB : channel(dy))
played_by A
def=
   local
   State : nat,
```

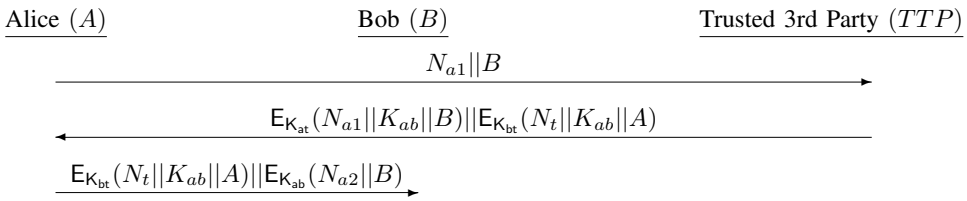

Fig. 4: Messages passed for unilateral authentication of Alice to Bob involving a TTP

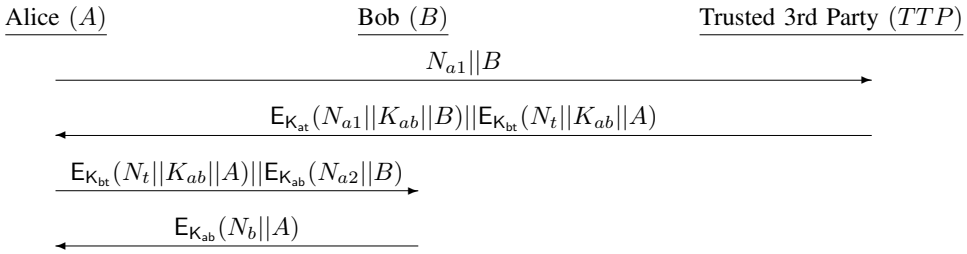

Fig. 5: Messages passed for mutual authentication of Alice and Bob involving a TTP

```
   K,Kbt : symmetric_key,
   Na1,Na2,Nt: text,
   X: {text.symmetric_key.agent}_symmetric_key
init
   State := 10
transition
   1. State = 10 /\ RB(start) =|>
   State':= 11 /\ Na1':= new()
   /\ ST(Na1'.B)
   2. State = 11
   /\ RT({Na1.K'.B}_Kat.X') =|>
   State':= 12 /\ Na2' := new()
   /\ SB(X'.{Na2'.B}_K')
   /\ witness(A,B,bob_alice_na2,Na2')
   /\ request(A,TTP,alice_ttp_na1,Na1)
end role
```

In addition to correct specification of roles, true identification of security goals is also very important in order to achieve reliable analysis of any security protocol. For unilateral authentication involving TTP, we identified and used 3 security goals and analysed the protocol against these specified goals. The goal section of our HLPSL specification is copied below:

```
goal
   secrecy_of k
   authentication_on bob_alice_na2
   authentication_on alice_ttp_na1
end goal
```

Two of the security goals are related to authentication while one is related to secrecy. The goal "secrecy_of k" is related to the secrecy of symmetric session key generated by the trusted third party and to be used by Alice and Bob for their communication. The second goal "authentication_on bob_alice_na2" is for authentication of Alice by Bob while the third goal "authentication_on alice_ttp_na1" is related to authentication of TTP by Alice.

For protocol analysis using AVISPA, an important consideration is the number of parallel sessions to be run among different protocol participants. Below is an excerpt from our HLPSL code which contains the *environment* role.

```
role environment()
def=
   const
   bob_alice_na2, alice_ttp_na1,
   k : protocol_id,
   kat,kbt,kit : symmetric_key,
   a,b,t,i : agent
intruder_knowledge = {a,b,t,i,kit}
composition
         session(a,b,t,kat,kbt)
      /\ session(a,i,t,kat,kit)
      /\ session(i,b,t,kit,kbt)
      /\ session(a,b,i,kat,kbt)
end role
```

In environment role above, four parallel sessions are shown. The first one is a protocol run among honest parties namely Alice, Bob and TTP while in other three sessions, the intruder masquerades as Bob, Alice and TTP, respectively.

When we analysed our specification against the specified goals using OFMC and CL-AtSe, both these AVISPA back-ends showed that the protocol is unsafe. The detected attack is an impersonation attack where the intruder tricks Bob into believing that the last message of the protocol (Figure 4) is coming from Alice. The intruder creates this last message by concatenating two message parts. The first part is the one which was earlier sent by the TTP to Alice and which is now replayed by the intruder. The second part of the message is the legitimate one which was sent by Alice intended for Bob. Figure 6 shows a *Message Sequence Chart* (MSC) of the attack detected using OFMC back end. The MSC is generated by using SPAN [23] which is a freely available protocol animation tool for AVISPA.

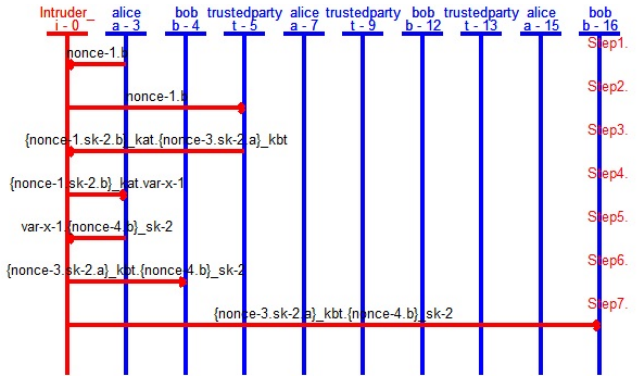

Fig. 6: An MSC of the replay attack detected by AVISPA against ISO/IEC 9798-2 unilateral authentication involving TTP. The attack is detected by OFMC back end and the MSC is generated by using the SPAN tool.

In step 6 of Figure 6, Bob (b-4) is not able to detect the modifications made by the intruder in the message coming from Alice and he is tricked to believe that the message he receives is coming unmodified from Alice. This violates one of the authentications goals (authentication_on bob_alice_na2) mentioned in the goal section of our HLPSL specification. Some explaination of this attack corresponding to MSC in Figure 6 goes as follows:

Step 1. Alice wants to initiate a session with Bob and sends a message containing Bob's identity and a nonce to the TTP.

Step 2. The intruder views the message and passes it to the TTP.

Step 3. The TTP replies with a message containing two parts, one intended for Alice and the other for Bob both containing the sessions key sk-2 in a digital envelope.

Step 4. The intruder intercepts the message, sends the part intended for Alice as it is but replaces the other part by any random string. As the part for Bob was encrypted using the shared secret key between the TTP and Bob, Alice has no way to differentiate between the legitimate encrypted value and the random string.

Step 5. Upon receiving the message, Alice decrypts the first part of the received message to retrieve the key sk-2 and prepares an encrypted message for Bob using sk-2. She also extracts the other part of received message and concatenates it with the before-mentioned encrypted message and sends it to Bob.

Step 6. The intruder intercepts the message sent by Alice and constitutes a new message by concatenating part of two messages, one which was sent by the TTP in step 3 and the other which was sent by Alice in step 5. The intruder sends this new message to Bob. Bob receives the message, decrypts first part using K_{bt} to get sk-2 and decrypt second part using sk-2 to get $nonce$-4 and b. Now Bob believes that he is talking to Alice and that they both agree on key sk-2 for communications. This is clearly a violation of the unilateral authentication guarantee of the protocol.

When the same protocol was analysed using CL-Atse back end, a similar attack against authentication goal was detected. Though the attack detected by CL-Atse is more complex and involves more steps as compared to the one detected by OFMC, the overall violation is the same. The MSC of the attack found by CL-Atse is shown in Figure 7. The message received by Bob (b-4) in step 10 is constituted by the intruder by combining parts from two earlier messages sent by the TTP and Alice in steps 7 and 9, respectively. Bob is not able to detect this modification and believes that he is receiving the message from Alice. Note that as mutual authentication involving TTP protocol (Figure 5) is an extension of unilateral authentication protocol (Figure 4) with one additional message, the above-mentioned attack is also applicable against mutual authentication protocol.

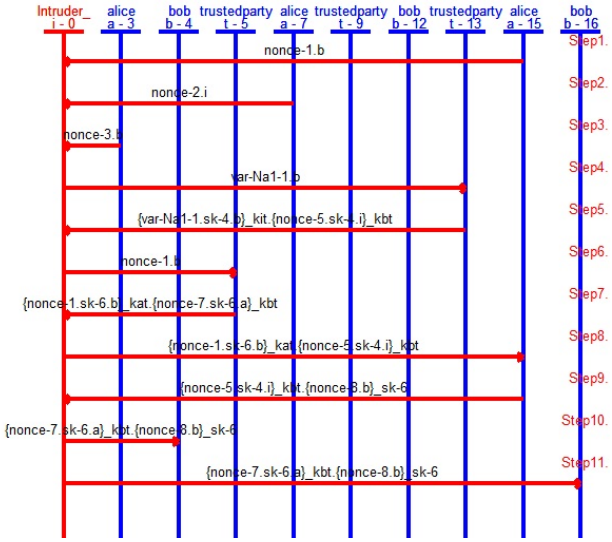

Fig. 7: An MSC of the attack detected by CL-AtSe back end of AVISPA against ISO/IEC 9798-2 unilateral authentication involving TTP.

In terms of efficiency, CL-AtSe is much faster than OFMC in all cases. The time consumed by these sub-tools is tabulated in Table II. The time is computed on an Intel Core i3 processor with 3 GB of RAM and running Microsoft Windows 7 Professional with Service Pack 1.

	Not Involving TTP		Involving TTP	
	Unilateral	Mutual	Unilateral	Mutual
OFMC	0.01	0.04	0.22	0.28
CL-ATSE	0	0.01	0.02	0.03

TABLE II: Running Time (in seconds) for ISO-IEC 9798-2 protocols using OFMC and CL-AtSe back-ends.

V. Conclusions

In this paper, we have performed formal analysis of ISO/IES 9798-2 entity authentication standard using AVISPA tool. We used OFMC and CL-AtSe back-ends for automated security analysis. Both back-ends found attacks against unilateral and mutual authentication protocols involving a trusted third party. In terms of efficiency, we found that CL-AtSe was more efficient in analyzing the protocols as compared to OFMC.

References

[1] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[2] D. Otway and O. Rees, "Efficient and timely mutual authentication," *Operating Systems Review*, vol. 21, no. 1, pp. 8–10, 1987.

[3] L. Gong, "Using one-way functions for authentication," *Computer Communication Review*, vol. 19, no. 5, pp. 8–11, 1989.

[4] S. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, may 1992, pp. 72 –84.

[5] C. Boyd, "A class of flexible and efficient key management protocols," in *9th IEEE Computer Security Foundations Workshop*, 1996.

[6] L. Fan, J. Li, and H. Zhu, "An enhancement of timestamp-based password authentication scheme," *Computers and Security*, vol. 21, no. 7, pp. 665–667, 2002.

[7] T. Chen, W.-B. Lee, and G. Horng, "Secure SAS-like password authentication schemes," *Computer Standards and Interfaces*, vol. 27, no. 1, pp. 25–31, 2004.

[8] S. W. Lee, H. S. Kim, and K. Y. Yoo, "Improved efficient remote user authentication scheme using smart cards," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 565–567, 2004.

[9] M. Hwang, C. Lee, and Y. Tang, "A simple remote user authentication scheme," *Mathematical and Computer Modelling*, vol. 36, no. 1, pp. 103–107, 2002.

[10] E. Yoon, E. Ryu, and K. Yoo, "An improvement of hwang-lee-tangs simple remote user authentication scheme," *Computers and Security*, vol. 24, no. 1, pp. 50–56, 2005.

[11] C. Li, C. Lee, and L. Wang, "A two-factor user authentication scheme providing mutual authentication and key agreement over insecure channels," *Journal of Information Assurance and Security*, vol. 5, no. 1, pp. 201–208, 2010.

[12] S. Kalsoom and S. Ziauddin, "Cryptanalysis and improvement of a two-factor user authentication scheme providing mutual authentication and key agreement over insecure channels," *International Journal of Machine Learning and Computing*, vol. 2, no. 5, 2012.

[13] H. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," in *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on*. IEEE, 1994, pp. 122–136.

[14] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 426, no. 1871, pp. 233–271, 1989.

[15] G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using FDR," *Software - Concepts and Tools*, vol. 17, no. 3, pp. 93–102, 1996.

[16] J. Mitchell, M. Mitchell, and U. Stern, "Automated analysis of cryptographic protocols using Mur-phi," in *IEEE Symposium on Security and Privacy*, 1997, pp. 141–151.

[17] D. X. Song, S. Berezin, and A. Perrig, "Athena: A novel approach to efficient automatic security protocol analysis," *Journal of Computer Security*, vol. 9, no. 1-2, pp. 47–74, 2001.

[18] C. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in *Computer Aided Verification*, 2008.

[19] D. Basin and C. Cremers, "Evaluation of ISO/IEC 9798 protocols version 2.0," Technical report, Cryptrec, Tech. Rep., 2011.

[20] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani *et al.*, "The AVISPA tool for the automated validation of internet security protocols and applications," in *17th International Conference on Computer Aided Verification (CAV)*, 2005.

[21] L. Vigano, "Automated security protocol analysis with the AVISPA tool," *Electronic Notes in Theoretical Computer Science*, vol. 155, no. 0, pp. 61–86, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066106001897

[22] S. International Organization for Standardization, Geneve, *9798-2:2008, Information technology - Security techniques - Entity Authentication - Part 2: Mechanisms using symmetric encipherment algorithms*, International Organization for Standardization Std., 2008.

[23] Y. Boichut, T. Genet, Y. Glouche, and O. Heen, "Using animation to improve formal specifications of security protocols," in *2nd Conference on Security in Network Architectures and Information Systems (SARSSI 2007)*, 2007, pp. 169–182.